

When the computer asks "Are you a robot?", maybe, he just wants to find his family?



ASSIGNMENT 2

ITRI 626

ENRICO DREYER

31210783

Table of Contents

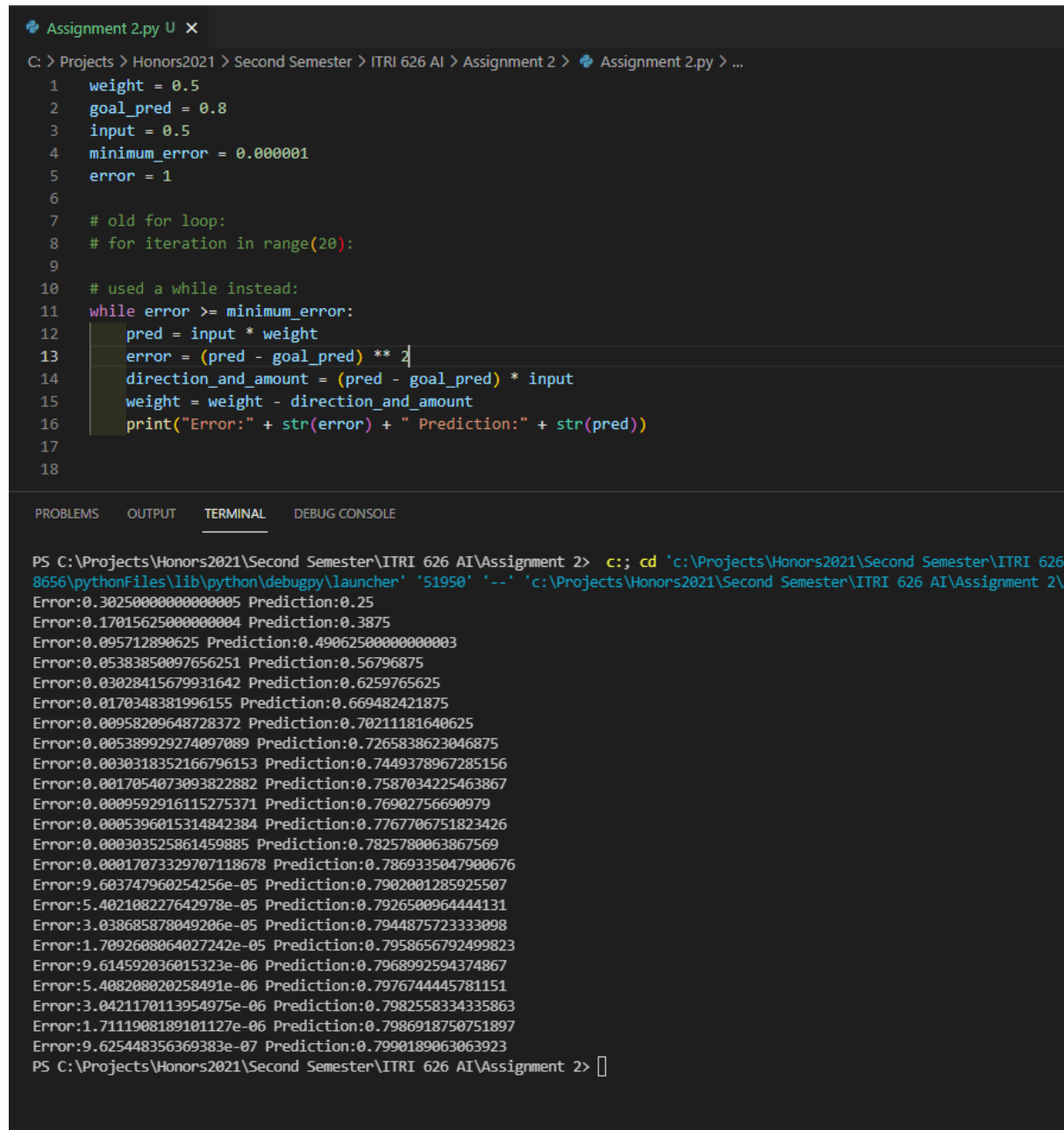
Modifying the code	2
Is there a better way?	2
References	5

Table of Figures

Figure 1: My code	2
Figure 2: Plot Graph	3
Figure 3: My second code snippet	4
Figure 4: Example code (Gau, 2021)	5

Modifying the code

After reading the assignment, I added a `minimum_error` variable that is equal to 0.000001. I used a `while` instead of a `for` loop to stop when the error is smaller than the minimum error. As seen in the screenshot below.



```
Assignment 2.py U X
C: > Projects > Honors2021 > Second Semester > ITRI 626 AI > Assignment 2 > Assignment 2.py > ...
1 weight = 0.5
2 goal_pred = 0.8
3 input = 0.5
4 minimum_error = 0.000001
5 error = 1
6
7 # old for loop:
8 # for iteration in range(20):
9
10 # used a while instead:
11 while error >= minimum_error:
12     pred = input * weight
13     error = (pred - goal_pred) ** 2
14     direction_and_amount = (pred - goal_pred) * input
15     weight = weight - direction_and_amount
16     print("Error:" + str(error) + " Prediction:" + str(pred))
17
18

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 2> c:; cd 'c:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 2\
8656\pythonFiles\lib\python\debugpy\launcher' '51950' '--' 'c:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 2\
Error:0.30250000000000005 Prediction:0.25
Error:0.17015625000000004 Prediction:0.3875
Error:0.095712890625 Prediction:0.49062500000000003
Error:0.05383850097656251 Prediction:0.56796875
Error:0.03028415679931642 Prediction:0.6259765625
Error:0.0170348381996155 Prediction:0.669482421875
Error:0.00958209648728372 Prediction:0.70211181640625
Error:0.005389929274097089 Prediction:0.7265838623046875
Error:0.0030318352166796153 Prediction:0.7449378967285156
Error:0.0017054073093822882 Prediction:0.7587034225463867
Error:0.0009592916115275371 Prediction:0.76902756690979
Error:0.0005396015314842384 Prediction:0.7767706751823426
Error:0.000303525861459885 Prediction:0.7825780063867569
Error:0.00017073329707118678 Prediction:0.7869335047900676
Error:9.603747960254256e-05 Prediction:0.7902001285925507
Error:5.402108227642978e-05 Prediction:0.7926500964444131
Error:3.038685878049206e-05 Prediction:0.7944875723333098
Error:1.7092608064027242e-05 Prediction:0.7958656792499823
Error:9.614592036015323e-06 Prediction:0.7968992594374867
Error:5.408208020258491e-06 Prediction:0.7976744445781151
Error:3.0421170113954975e-06 Prediction:0.7982558334335863
Error:1.7111908189101127e-06 Prediction:0.7986918750751897
Error:9.625448356369383e-07 Prediction:0.7990189063063923
PS C:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 2> ]
```

Figure 1: My code

Is there a better way?

At first, I thought that the assignment can not be that easy. But as I read the assignment in detail, I understood that the main objective is to explain that in practice this is not a good idea.

I started by reading the *Grokking Deep Learning Textbook* from page 65 (Trask, 2019).

And the first topic I read was on understanding the relationship between the weight and the error, and that one variable changes the other. This was hot and cold learning that we used in assignment 1.

Then they explained the “Box with rods”, whereas you change one rod, the other rod will move with the first rod, this explained in simple terms what derivatives are. “When you move x, how much does y move”. In this assignment, the *goal_pred* and *input* are fixed we can look at the derivative as a slope, this explains how much does one variable change, when you change the other.

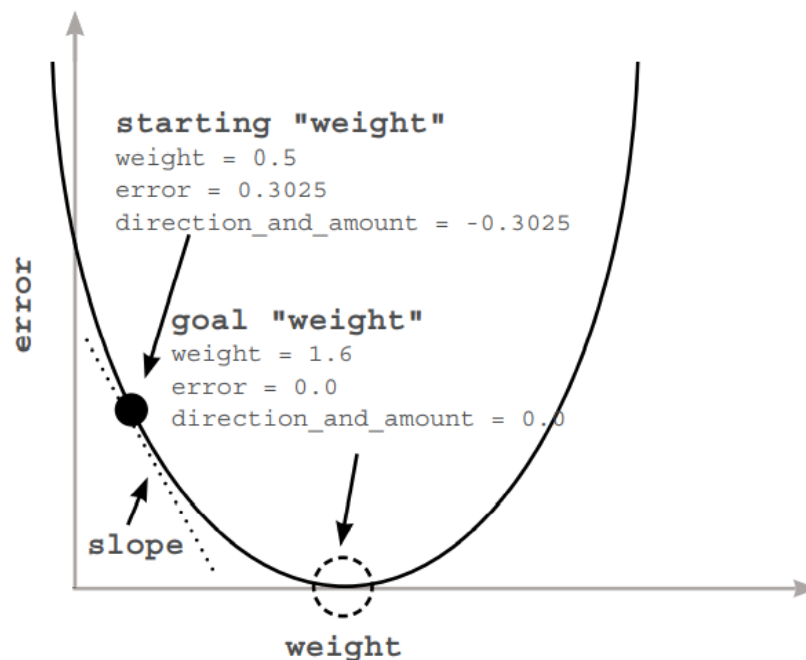


Figure 2: Plot Graph

When you use derivatives, you can pick two variables in a formula, and know how they interact with each other. And when I thought about it, a neural network is a couple of weights that you can use to compute an error function.

The textbook also explained a better way, and that is to use derivatives. The slope in the graph represents the derivative. The slope points to the opposite direction to the lowest point of the curve. Thus, if you have a negative slope, you can increase the weight to get the minimum errors.

A derivative gives the relationship between the weight and the error, so you can move the weight in the opposite direction of the derivative to find the lowest weight. This method is called Gradient Descent, and this is one of the better ways to do the assignment, as the neural network learns as it goes.

```
Assignment 2.py U X
C: > Projects > Honors2021 > Second Semester > ITRI 626 AI > Assignment 2 > Assignment 2.py > ...
1 weight = 0.5
2 goal_pred = 0.8
3 input = 0.5
4 minimum_error = 0.000001
5 error = 1
6 weight_delta = 0
7 delta = 0
8 count = 0
9
10 # old for loop:
11 # for iteration in range(20):
12
13 # used a while instead:
14 while error >= minimum_error:
15     pred = input * weight
16     error = (pred - goal_pred) ** 2
17
18     # direction_and_amount = (pred - goal_pred) * input
19     # weight = weight - direction_and_amount
20
21     delta = pred - goal_pred
22
23     #this is the derivative
24     weight_delta = delta * input
25
26     weight = weight - weight_delta
27     count += 1
28
29     print("Iteration:" + str(count) + " Error:" + str(error) + " Prediction:" + str(pred))
30
31
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Iteration:7 Error:0.00958209648728372 Prediction:0.70211181640625
Iteration:8 Error:0.005389929274097089 Prediction:0.7265838623046875
Iteration:9 Error:0.0030318352166796153 Prediction:0.7449378967285156
Iteration:10 Error:0.0017054073093822882 Prediction:0.7587034225463867
Iteration:11 Error:0.0009592916115275371 Prediction:0.76902756690979
Iteration:12 Error:0.0005396015314842384 Prediction:0.7767706751823426
Iteration:13 Error:0.000303525861459885 Prediction:0.7825780063867569
Iteration:14 Error:0.00017073329707118678 Prediction:0.7869335047900676
Iteration:15 Error:9.603747960254256e-05 Prediction:0.7902001285925507
Iteration:16 Error:5.402108227642978e-05 Prediction:0.7926500964444131
Iteration:17 Error:3.038685878049206e-05 Prediction:0.7944875723333098
Iteration:18 Error:1.7092608064027242e-05 Prediction:0.7958656792499823
Iteration:19 Error:9.614592036015323e-06 Prediction:0.7968992594374867
Iteration:20 Error:5.408208020258491e-06 Prediction:0.7976744445781151
Iteration:21 Error:3.0421170113954975e-06 Prediction:0.7982558334335863
Iteration:22 Error:1.711908189101127e-06 Prediction:0.7986918750751897
Iteration:23 Error:9.625448356369383e-07 Prediction:0.7990189063063923
PS C:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 2> 
```

Figure 3: My second code snippet

I started searching ways to create a simple neural network, to give me insight on different networks and what they do to achieve what is asked in the assignment (Gau, 2021). They also used a derivative to solve the problem, as shown in the example below. This gave me insight on how the neural network decides how to manipulate the weight in the right direction, as shown in the snippet below.

The difference between an optimal neural network and an imperfect neural network is the total iterations that it takes for the network to find its goal.

```
# A simple neural network class
class SimpleNN:
    def __init__(self):
        self.weight = 1.0
        self.alpha = 0.01

    def train(self, input, goal, epochs):
        for i in range(epochs):
            pred = input * self.weight
            delta = pred - goal
            error = delta ** 2
            derivative = delta * input
            self.weight = self.weight - (self.alpha * derivative)
            print("Error: " + str(error))

    def predict(self, input):
        return input * self.weight
```

Figure 4: Example code (Gau, 2021)

References

Gau, L. (2021). How To Create A Simple Neural Network Using Python.

<https://hackernoon.com/how-to-create-a-simple-neural-network-using-python-6o2d33yo>

Trask, A. W. (2019). *Grokking Deep Learning*